**System control and diagnostic protocol**

This protocol is meant for medium speed debugging and control of systems via a RS-485 physical layer and uses UART protocol as the network layer. The network consists of a single client and a single server. The protocol supports two modes of transfer: standard frame transfer and fast transfer. Standard frame transfer can be used to transfer data at a lower rate (6 bytes/frame) at any sequence. Fast transfer allows the transfer of information in a pre-defined sequence determined by exchange of message control frames between the client and server.

**Standard frame transfer mode (SFT)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| BYTE [5] | | BYTE [4] | BYTE [3] | BYTE [2] | BYTE [1] | BYTE [0] |
| BIT [7] | BIT [6:0] |  |  |  |  |  |
| PRI | FID | DB [3] | DB [2] | DB [1] | DB [0] | CKSUM8 |

|  |  |  |
| --- | --- | --- |
| PRI | Priority | This bit sets the priority of the message frame. A value of 0 means low-priority and 1 means high-priority. When a server receives data from the client, it stores the frames in a circular queue. When priority is high, the received frame is queued to the front and when the priority is low, the received frame is queue at the back of the queue. |
| FID | Frame ID | The 8-bit frame ID. Each message frame is identified by a frame ID. |
| DB [3] to DB [0] | Data bytes | Data bytes containing information. DB [3] is the most significant byte and DB [0] is the least significant byte. |
| CKSUM8 | Checksum | The 8-bit checksum calculated for the data frame calculated for bytes BYTE [5] to BYTE [1]. |

The values of the frame IDs (FID) can be from 0 to 127 when PRI = 0 and 0 to 120 when PRI = 1. The frame IDs reserved for 120 to 127 when PRI = 1 are reserved for controlling message flow such as changing modes of message transfer. The minimum communication for the communication of a message frame is given by:

For example, when communicating over the RS-485 bus using UART with 1 stop bit and no parity enabled at a baud rate of 230400, the baud rate is given by:

**Message control frame**

Message control frames are used to control the mode of transfer of information between the client and server. The message control frames are as follows.

|  |  |
| --- | --- |
| PRI = 1 and FID = 120 | transfer on request control frame 1 (FTOR1) |
| PRI = 1 and FID = 121 | transfer on request control frame 2 (FTOR2) |
| PRI = 1 and FID = 122 | Block transfer request frame 1. |
| PRI = 1 and FID = 123 | Block transfer request frame 2. |
| PRI = 1 and FID = 124 | Block transfer request frame 3. |
| PRI = 1 and FID = 125 | Transfer mode control frame. |

*[1] Transfer on request frames.*

This control frame is usually sent from the client to server to request for a certain message frame. The client requests the server to send message frame of a certain ID (FTOR\_IDx) and waits for an acknowledgement from the server side (FTOR\_ACK). Once the server has acknowledged the message frame(s), it has to send the requested message frame(s) to the client. The number of message frames which can be transferred on requested should be specified by FTOR\_NF in FTOR1 message frame. The requested IDs are specified by FTOR\_ID0 to FTOR\_ID2 in FTOR1 and FTOR\_ID3 to FTOR\_ID6 in FTOR2. When requesting for multiple message frames on request, if the number of requested frames exceeds 3, FTOR2 must be sent followed by FTOR1.

FTOR1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BYTE [5] | | | BYTE [4] | | | BYTE [3] | BYTE [2] | BYTE [1] | BYTE [0] |
| BIT [7] | BIT [6:0] | | BIT [7] | BIT [6] | BIT [5:0] |  |  |  |  |
| 1 | | 120 | FTOR\_ACK | FTOR\_REQ | FTOR\_NF | FTOR\_ID0 | FTOR\_ID1 | FTOR\_ID2 | CKSUM8 |

FTOR2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| BYTE [5] | | | BYTE [4] | BYTE [3] | BYTE [2] | BYTE [1] | BYTE [0] |
| BIT [7] | BIT [6:0] | |  |  |  |  |  |
| 1 | | 121 | FTOR\_ID3 | FTOR\_ID4 | FTOR\_ID5 | FTOR\_ID6 | CKSUM8 |

*[2] Block transfer frames.*

Block transfer mode is used to transfer a single parameter or multiple-parameters between the client and server at a higher bandwidth in a pre-defined sequence which is specified by the client to the server. The client initiates the block transfer request by sending a block transfer request (BTR\_REQ) and specifying the number of times the sequence of parameters (BTR\_NTR) and the number of parameters per sequence (BTR\_NPAR) which have to be sent by the server. The client acknowledges the request for transfer by raising the BTR\_ACK flag. Each parameter is identified by a unique parameter ID (PID). These parameter IDs should be defined in software from both client and server side. For block transfer, there are three frames, BTR1, BTR2 & BTR3. When requesting for transfer of more than one parameter, BTR2 and BTR3 should be sent followed by BTR1.

BTR1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BYTE [5] | | BYTE [4] | | | BYTE [3] | BYTE [2] | BYTE [1] | BYTE [0] |
| BIT [7] | BIT [6:0] | BIT [7] | BIT [6] | BIT [5:0] |  |  |  |  |
| 1 | 122 | BTR\_REQ | BTR\_ACK | BTR\_NPAR | BTR\_NTRH | BTR\_NTRL | BTR\_PID0 | CKSUM8 |

BTR2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| BYTE [5] | | BYTE [4] | BYTE [3] | BYTE [2] | BYTE [1] | BYTE [0] |
| BIT [7] | BIT [6:0] |  |  |  |  |  |
| 1 | 123 | BTR\_PID1 | BTR\_PID2 | BTR\_PID3 | BTR\_PID4 | CKSUM8 |

BTR3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| BYTE [5] | | BYTE [4] | BYTE [3] | BYTE [2] | BYTE [1] | BYTE [0] |
| BIT [7] | BIT [6:0] |  |  |  |  |  |
| 1 | 124 | BTR\_PID5 | BTR\_PID6 | BTR\_PID7 | BTR\_PID8 | CKSUM8 |

[3] Transfer mode control frame.

This frame is used to control the mode of message transfer. This allows the client to enable and disable message transfer with the client.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BYTE [5] | | BYTE [4] | | | | | BYTE [3] | | | | | | | | | BYTE [2] | | BYTE [1] | | BYTE [0] |
| BIT [7] | BIT [6:0] | BIT [7:4] | BIT [3] | BIT [2] | BIT [1] | BIT [0] | BIT [7:6] | BIT [5] | BIT [4] | | BIT [3] | BIT [2] | | BIT [1] | BIT [0] | BIT [7:1] | BIT [0] | BIT [7:1] | BIT [0] |  |
| 1 | 126 | 0 | TMS\_REQ | TMS\_ACK | TMC\_REQ | TMC\_ACK | 0 | BTR\_ENSTAT | FTOR\_ENSTAT | SFT\_ENSTAT | | | BTR\_ENREQ | FTOR\_ENREQ | SFT\_ENREQ | 0 | MTR\_ENREQ | 0 | MTR\_ENACK | CKSUM8 |

Symbol descriptions

|  |  |  |
| --- | --- | --- |
| TMS\_REQ | Transfer mode status request | Setting this bit requests the server to send its current mode of data transfer |
| TMS\_ACK | Transfer mode status request acknowledgement | The server sets this bit indicating that it has processed the transfer mode status request. |
| TMC\_REQ | Transfer mode change request | Setting this bit requests the server to change its current mode of data transfer. |
| TMC\_ACK | Transfer mode change request acknowledgement | The server sets this bit indicating that it has processed the transfer mode change request. |
| BTR\_ENSTAT | Block transfer enable status | This bit is set/reset based on whether the server is in block transfer mode or not. |
| FTOR\_ENSTAT | Frame transfer on request enable status | This bit is set/reset based on whether the server is in frame transfer on request mode or not. |
| SFT\_ENSTAT | Standard frame transfer enable status | This bit is set/reset based on whether the server is in standard transfer mode or not. |
| BTR\_ENREQ | Block transfer enable request | This bit is set/reset by the client to request to change the transfer mode to block transfer mode. |
| FTOR\_ENREQ | Frame transfer on request enable request | This bit is set/reset by the client to request to change the transfer mode to frame transfer on request mode. |
| SFT\_ENREQ | Standard frame transfer enable request | This bit is set/reset by the client to request to change the transfer mode to standard frame transfer mode. |
| MTR\_ENREQ | Message transfer enable request | This bit is enabled by client to enable message transfer between client and server. |
| MTR\_ENACK | Message transfer acknowledgement | This bit is enabled by server to indicate message transfer between client and server is active. |

\*Highlighted parts are not implemented in protocol stack.